Computing and Informatics, Vol. 26, 2007, 171–197

KNOWLEDGE DISCOVERY IN DATABASE: INDUCTION GRAPH AND CELLULAR AUTOMATON

Baghdad Atmani

Leibniz Laboratory, IMAG
46 Av Felix Viallet, 38031 Cedex Grenoble, France
&
Department of Computer Science, Faculty of Science, University of Oran,
BP 1524 El M'Naouer 31000 Oran, Algeria
e-mail: baghdad.atmani@imag.fr

Bouziane Beldjilali

Department of Computer Science, Faculty of Science, University of Oran,
BP 1524 El M'Naouer 31000 Oran, Algeria
e-mail: bouziane.beldjilali@univ-oran.dz

Manuscript received 8 June 2006; revised 19 December 2006
Communicated by Prabhat Kumar Mahanti

Abstract. In this article we present the general architecture of a cellular machine, which makes it possible to reduce the size of induction graphs, and to optimize automatically the generation of symbolic rules. Our objective is to propose a tool for detecting and eliminating non relevant variables from the database. The goal, after acquisition by machine learning from a set of data, is to reduce the complexity of storage, thus to decrease the computing time. The objective of this work is to experiment a cellular machine for systems of inference containing rules. Our system relies upon the graphs generated by the SIPINA method.

After an introduction aiming at positioning our contribution within the area of machine learning, we briefly present the SIPINA method for automatic retrieval of knowledge starting from data. We then describe our cellular system and the phase of knowledge post-processing, in particular the validation and the use of extracted knowledge.

The presentation of our system is mostly done through an example taken from medical diagnosis.

Keywords: Symbolic system, induction graph, automatic training, cellular automaton, rule extraction, medical diagnosis

1 INTRODUCTION

Many medical organizations collect and manage increasingly large and bulky masses of information [8, 18, 21, 27, 30]. This increasing number of large databases are seldom exploited for extracting new knowledge on various phenomena or simply to clarify decisions [14]. Knowledge discovery from database (KDD) [9, 14, 36] is a recent concern in data-processing research. In addition to the discovery of new knowledge, the techniques of knowledge extraction [12, 17, 19, 20, 24] can contribute to the implementation of expert systems, by reducing the cost and the difficulty of traditional techniques of knowledge acquisition from human experts.

A solution to this problem is to design data-processing programs able to learn and discover their own knowledge starting from practical examples [6]. In this type of systems, the expertise is not provided any more by human experts, but is built starting from data relating to the field. Knowledge extraction starting from data relies upon the principles of machine learning [2, 25, 26] and uses supervised inductive training methods [4, 19, 20]. Among these methods, we are interested more particularly in those which are based on induction graphs [11, 15], because the classification function is expressed by graphs which can be transformed into production rules. However, the direct use of the rules extracted from an induction graph [27, 28, 36] is not possible, for several reasons. On the one hand, production rules are generally conjunctive and not disjunctive-conjunctive like those resulting from an induction graph. In addition, since variables can appear several times in the graph, the graphs obtained can then be of a large size and comprise redundant and incoherent information. It is then necessary to simplify the graphs generated, and consequently the outcoming productions rules [3, 7, 27, 36].

Within the framework of a medical project, we have been submitted a problem concerning diabetic patients, with the purpose of designing an automatic system for identification of the diabetes types. In order to propose a first automatic model of prediction of the various diabetes types, we tried knowledge extraction using several methods of KDD [19, 20, 22, 36]. We used, the regression model, discriminating analysis, a multi-layer perceptron, four methods relying upon induction graphs (ID3, C4.5, CART, SIPINA) and the system CHARADE [10]. Through a comparison of the results of the various methods, it appears that induction graphs, and in particular the SIPINA method [36], show better rates of classification with a low number of rules.

Our contribution within the framework of this real application is twofold. On the one hand, we propose a new method of representation and simplification of graphs generated by the SIPINA method, with the detection and the elimination of useless splitting-fusion parts. In addition, to feed the knowledge base of an expert system,

we generate simple conjunctive rules starting from the optimal induction graph. In response to the limitations of various approaches [3, 7, 27] to rules simplifications, we propose our own cellular machine which makes it possible to eliminate redundant and incoherent information in order to produce an optimal set of rules.

2 CHOSEN APPROACH FOR AUTOMATIC TRAINING

All along this paper, we take as an example the assistance to medical diagnosis. One can consider two approaches [36]:

1. Training by heart: the knowledge that describes what has to be done in the presence of some symptoms is introduced in the machine in the form of rules collected from physicians. The system is endowed with an inference engine that allows to use this knowledge in order to infer new facts.
2. Automatic training: in this case, knowledge is not provided by experts but generated by the machine from already encountered situations. This is training or automatic learning from examples or from data.

The problem with automatic learning from examples is to let this process induce only correct classification procedures from correctly classified data. For example, from a set of data about patients treated for diabetes, correctly stored by physicians, the machine will try to determine diagnosis rules that are applicable to new patients in order to determine if they are insulin dependent or not. Once validated, these rules can be inserted into a running system. Following [36], the knowledge produced by the machine, also called output of the automatic training, is not necessarily of a logical nature, and it can take various forms: neuron networks, algebraic models, geometric models, etc.

There are two main categories of techniques for achieving this [9, 14, 36]:

1. Symbolic techniques, based on knowledge of experts, in limited number, without errors and ready for logical calculation.
2. Symbolic techniques with numerical induction from data, that learn from a numerical representation of applicable information for training.

The approach that we propose here belongs to the second category. Using a cellular automaton [1] cooperating with an induction graph (SIPINA method), our cellular system permits to extract new knowledge from observed data. With the goal of extracting applicable knowledge that reflects the reality and that will be validated by experts of the domain, data or observations presented to the machine are supposed to contain sufficient information to be generalizable, and rich enough to cover as much cases as possible.

3 PRESENTATION FROM AN EXAMPLE

In a context of diabetic patients monitoring [8, 21, 27], setting up tools for accident detection is not possible without considering the necessary role that the physician must have. The aim is to design a system for assisted monitoring and diagnosis that will provide specialists with the necessary information for identifying the diabetes type of patients.

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_p\}$ be the population of diabetic patients taken into account for the training. An attribute is associated with this population, called endogenous variable (also called explicative variable or class attribute), denoted C.

A class $C(\omega)$ can be associated with every individual ω. The endogenous variable C takes its values in the set IC of class identifiers.

$$C : \Omega \to IC = \{c_1, c_2, \ldots, c_m\}$$
$$\omega_i \mapsto C(\omega_i) = c_j$$

In the example, the aim is diagnosing a possible insulin dependence. This will be designated by an endogenous variable $C : \Omega \to \{1, 2\}$, where class 1 contains all insulin dependent patients (*diabetes of type I*), and class 2 the non dependent patients (*diabetes of type II*). The objective is to define a function φ for predicting the class C, thus the detection of possible accidents. The determination of the prediction model φ, which is the goal of the training, is bound to the hypothesis that the values taken by the endogenous variable C are not at random, but depend upon certain individual situations, called exogenous variables that are determined by the expert.

The exogenous variables concerning an individual constitute a tuple of attributes:

$$X = (X_1, X_2, \ldots, X_n).$$

The exogenous variables take their values in a set IM of mode identifiers:

$$X : \Omega \to IM$$

$$X(\omega) = (X_1(\omega), X_2(\omega), \ldots, X_n(\omega)).$$

The value taken by $X_j(\omega)$ is called the modality of the attribute X_j for ω. In our case the exogenous variables are summarized in Table 1.

Updating φ requires two samples denoted Ω_a and Ω_t, which are subsets of Ω. The first one, Ω_a, used for training, will serve for the construction of φ. The second one, used for test, will serve for testing the validity of φ. For all patients $\omega \in (\Omega_a \cup \Omega_t)$ we assume that both the values $X(\omega)$ and the class $C(\omega)$ are known.

We also define Ω_e, the set of individuals in Ω_t (patients) not correctly classified during the test of the symbolic training.

Exogenous Var	**Semantics**	**Values**
X_1	Seniority	> 35; ≥ 15 and < 30; unspecified
X_2	How revealed	Spontaneous; Infectious; Glycemy unbalance; Recent
X_3	Weight	Normal; Skinny; Obese; Overweight
X_4	Viral Infection	Yes; No
X_5	State	Weight loss; No Weight loss
X_6	Association	Relation with auto-immune illness; No relation
X_7	Circumstance of discovery	Diabetic feet; Fortuitous; Infection; Retinopathy; Comas; Inaugural; Cetosic coma
X_8	Asthenia	Yes; No
X_9	Antecedent	Family; Personal; No Antecedent
X_{10}	Sex	Feminine, Masculine

Table 1. Exogenous variables, semantics and possible answers

4 GENERAL PROCESS OF TRAINING

The general process of training followed by our cellular system CASI (Cellular Automaton for System Inference) is organized in five stages:

1. Symbolic training by the SIPINA method;
2. Induction graph generation by the cellular automaton;
3. Induction graph optimization;
4. Conjunctive rules generation by the cellular automaton;
5. Validation by the cellular automaton.

Figure 1 summarizes the general diagram of our system.

4.1 Symbolic Training by the SIPINA Method

With the help of the sample Ω_a we start the symbolic treatment for the construction of the induction graph (SIPINA method) [35]:

1. Set the measure of uncertainty;
2. Set parameters: λ, μ and the initial partition S_0;
3. Apply the SIPINA algorithm for going from partition S_i to S_{i+1} and generate induction graph;
4. Generate prediction rules [27] [28].

The parameters λ, μ, the partitions and all other notions used in this process are introduced by means of examples and defined in the following paragraphs.

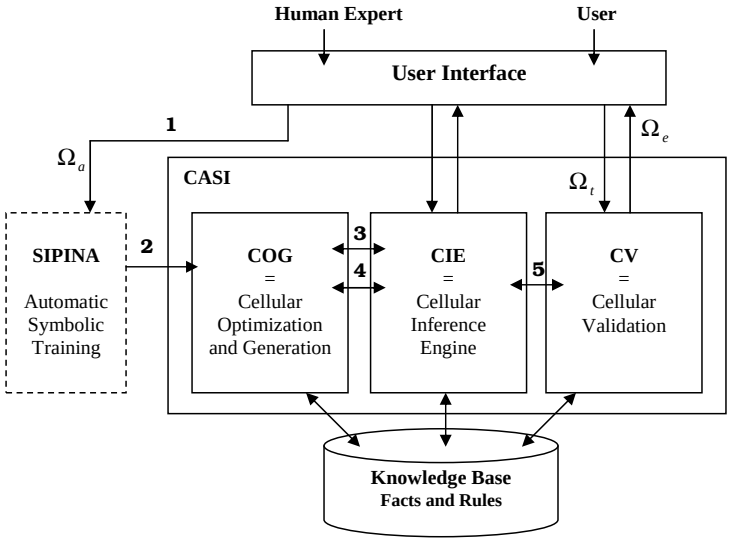


Fig. 1. General diagram of our system

4.1.1 Definition of a Partition Through an Example

The SIPINA method is a heuristic for the construction of a non arborescent induction graph [36]. Its principle consists in performing a succession of stages of fusion and/or splitting of nodes in the graph. Let us suppose that our training sample is composed of 15 diabetic patients belonging to two classes 1 and 2 (Table 2).

The initial partition S_0 includes only one element denoted s_0 that contains the sample, with 10 individuals belonging to class 1 and 5 belonging to class 2. The next partition $S_1 = (s_1, s_2, s_3)$ is generated by the variable X_1. The individuals in node s_i are defined as follows: $s_1 = \{\omega \in \Omega_a | X_1(\omega) = 0\}$, $s_2 = \{\omega \in \Omega_a | X_1(\omega) = 1\}$ and $s_3 = \{\omega \in \Omega_a | X_1(\omega) = 2\}$.

As in s_0, one distinguishes in s_1, s_2 and s_3 individuals of classes 1 and 2 respectively. Figure 2 summarizes the construction stages of s_0, s_1, s_2 and s_3.

From partition S_1, the process is iterated looking for a partition S_2 that would be better according to some criteria.

4.1.2 Measure of Quality of a Partition

The objective of the SIPINA method is to optimize a criteria τ_λ, called variation of uncertainty, during the transition from S_i to S_{i+1}, defined by $\Delta\tau_\lambda^{(i+1)} = \tau_\lambda^{(S_i)} - \tau_\lambda^{(Si+1)}$.

Let λ be a positive and non zero parameter. For the calculation of $\tau_\lambda^{(S_i)}$, one can use several functions constructed from uncertainty measures, like:

Ω_a	**CLASS**	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
ω_1	1	0	2	2	0	0	0	1	0	2	1
ω_2	1	2	2	0	1	0	0	1	0	2	0
ω_3	1	2	2	0	0	1	0	1	0	2	0
ω_4	1	0	2	0	0	1	0	1	0	2	0
ω_5	1	0	0	2	0	0	0	1	0	2	0
ω_6	1	0	2	0	0	0	0	1	0	2	0
ω_7	1	0	2	0	0	0	0	1	0	2	0
ω_8	1	0	2	0	0	0	0	1	0	2	1
ω_9	1	0	1	0	0	0	0	1	0	2	0
ω_{10}	1	0	1	0	0	0	0	1	0	2	0
ω_{11}	2	1	0	1	1	0	1	6	1	2	0
ω_{12}	2	1	3	1	1	0	1	6	1	2	0
ω_{13}	2	1	0	1	1	1	1	6	1	2	0
ω_{14}	2	1	2	0	0	1	0	6	0	2	0
ω_{15}	2	1	0	1	0	1	1	6	1	2	0

Table 2. Example of training sample

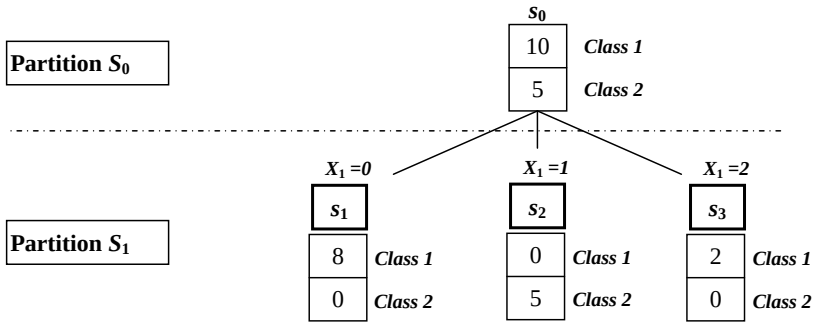


Fig. 2. The construction stages of s_0, s_1, s_2 and s_3

- the Shannon entropy:

$$\tau_\lambda^{(S_i)} = \sum_{j=1}^{K} \frac{n_{.j}}{n} \left(- \sum_{i=1}^{m} \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \log_2 \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \right)$$

- or the Quadratic entropy:

$$\tau_\lambda^{(S_i)} = \sum_{j=1}^{K} \frac{n_{.j}}{n} \left(- \sum_{i=1}^{m} \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \left(1 - \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \right) \right)$$

where:
n_{ij}: size of the population coming from class c_i, which is at node s_j;
$n_{i.}$: total size of the class c_i;

$n_{.j}$: total size of node s_j;
n: total size of Ω_a;
m: number of classes;
K: number of nodes s_j.

In our example, the distribution T_1 of individuals for $K = 3$ and $m = 2$ in partition S_1 (Table 2) is represented in Table 3. We can deduce, for $n_{11} = 8$ and $n_{21} = 0$, that the majority class of the node s_1 is c_1.

T_1	s_1	s_2	s_3	$Total$
c_1	$n_{11} = 8$	$n_{12} = 0$	$n_{13} = 2$	$n_{1.} = 10$
c_2	$n_{21} = 0$	$n_{22} = 5$	$n_{23} = 0$	$n_{2.} = 5$
$Total$	$n_{.1} = 8$	$n_{.2} = 5$	$n_{.3} = 2$	$n = 15$

Table 3. The distribution T_1

Setting the parameter λ. The parameter λ of the uncertainty measure controls the construction of the graph by penalizing partitions having many nodes with low population, thus favoring the fusion of such nodes.

The value of λ can be set arbitrarily to 1 or 2, but it can also be determined in an optimal way. The solution proposed by Zighed [36] is to define a node of low size. The value of λ is such that among all possible distributions T_k of μ individuals on m classes we have:

$$\lambda = \max\left(\lambda\,(m-1)\,\frac{2\mu^2 + 2\mu + m\lambda + 2\mu m\lambda}{(\mu + m\lambda)^2\,(\mu + 1 + m\lambda^2)}\right).$$

For a simple example $m = 2$ and $\mu = 2$, the optimal value will be $\lambda = 0,61098$.

Setting the value of μ. There exist two strategies to determine the minimal size μ of a node. The first one consists in asking the user to give the minimum number of individuals that every node should include. The second one consists in calculating this number while adopting a statistical view point.

The size of the training sample is $n = n_e + n_c$ where n_e is the number of individuals in the class of examples, and n_c the number of individuals in the class of counter examples. Let s be a terminal node of the induction graph whose total size is $n_s = n_{se} + n_{sc}$. The value of μ for a critical threshold $\alpha_0 = 0.05$ is then:

$$\mu = n_s = -\log(\alpha_0) \times \left(\frac{n}{n_c}\right)$$

4.1.3 How to Go from Partition S_i to S_{i+1}

Let us consider the example of Figure 2. Partition S_1 includes three elements s_1, s_2 and s_3. Going from partition S_1 to partition S_2 is done in three phases:

Fusion. One can note that from S_1 we can generate only one partition by fusion. This phase consists in gathering the nodes belonging to S_1 for generating only one node in S_2 while optimizing the criteria $\tau_\lambda^{(S_2)}$.

If the gain on the uncertainty $\tau_\lambda^{(S_2)} - \tau_\lambda^{(S_1)}$ is positive, then S_2 is generated. Otherwise, go to phase 2. Note that the fusion is always done between two nodes: if there were three nodes s_1, s_2 and s_3, three partitions could be generated by fusion (two by two) and we would choose the one that maximizes τ_λ.

Fusion-splitting. As in phase 1, fusions are done between all pairs of nodes. On every node produced by a fusion we search for the best admissible partition by splitting all variables X_i.

For example, with three nodes in S_1 and three variables, we generate three different partitions for each of the three nodes coming from the fusion in S_2, which gives nine possible partitions. Among all admissible partitions, we then keep those that lead to the best gain on the uncertainty.

Splitting. On every $s \in S_i$, we look for the best admissible partition by splitting all exogenous variables, and we keep the one that optimizes τ_λ.

Figure 3 summarizes the different phases.

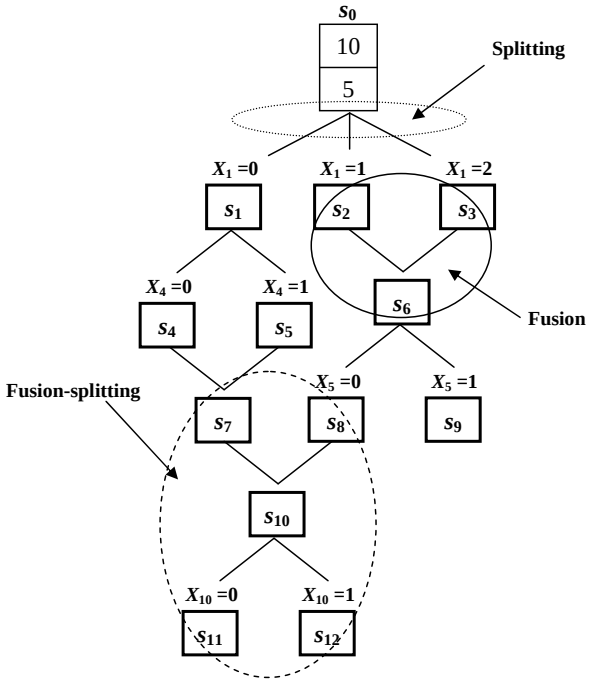


Fig. 3. Going from partition S_i to S_{i+1}

4.1.4 Generation of Rules

At the end of the symbolic treatment, we can generate the rules coming from the decision tree (graph). Let us consider the graph of Figure 3 as if it was a final induction graph, without worrying about checking the details of all calculations that lead to this graph. At that point, we can deduce three prediction rules R_1, R_2 and R_3 that have the form *if* **condition** *then* **conclusion**, where **condition** is a logical expression in disjunctive-conjunctive form and **conclusion** is the majority class in the node reached by the condition. For example, in Figure 2, the majority class of s_1 is 8 (class 1), but the majority class of s_2 is 5 (class 2).

R_1: `if` $((X_1 = 1)$ `or` $(X_1 = 2))$ `and` $(X_5 = 1)$ `then majority class of` s_9.

R_2: `if` $(((X_1 = 1)$ `or` $(X_1 = 2))$ `and` $(X_5 = 0)$ `and` $(X_{10} = 0))$ `or` $((X_1 = 0)$ `and` $((X_4 = 0)$ `or` $(X_4 = 1))$ `and` $(X_{10} = 0))$ `then majority class of` s_{11}.

R_3: `if` $(((X_1 = 1)$ `or` $(X_1 = 2))$ `and` $(X_5 = 0)$ `and` $(X_{10} = 1))$ `or` $((X_1 = 0)$ `and` $((X_4 = 0)$ `or` $(X_4 = 1))$ `and` $(X_{10} = 1))$ `then majority class of` s_{12}.

4.2 Induction Graph Generation by the Cellular Automaton

4.2.1 Definition of the Cellular Automaton [33, 34]

A cellular automaton is a grid of cells which change their state in discrete steps. After each step, the state of each cell is modified according to those of its neighbours before that step. The cells are updated in a synchronous way and the transitions are carried out, in theory, simultaneously [29]. By observing simple rules and specific transitions, a cellular automaton can carry out, in a global way, rather complex operations [5, 13, 23, 31, 32]. Some of the key concepts for cellular automata are the following:

Configuration: The global state of the cellular automaton is composed of the states of all its cells and is called the configuration of the cellular automaton.

Vicinity: The next state of each cell depends on the current state of its neighbours. The transition from one configuration of the automaton to the next is the consequence of the local transitions of all cells. The vicinity of a cell defines the set of its neighbours whose states are taken into consideration for each transition.

Parallelism: All the cells constituting the cellular automaton are updated in a simultaneous and synchronous way.

Determinism: For each cell, the new state is determined by the state of its vicinity only.

Homogeneity: All cells use the same transition rule to determine their next state.

Discretization: A cellular automaton behaves in discrete time steps from one state to the next.

A cellular automaton can be described by the following four components:

Dimension: There is no limit to dimensions, but in practice one uses automatons with 1, 2 or, 3 dimensions.

Vicinity of a cell: This defines the set of neighbour cells whose state will be taken into consideration for deciding on the next state of each cell.

State space: This is the finite set of states that a cell can be in.

Transition function: This is the set of rules which determine the new cell state according to its preceeding state and the preceeding states of the cells in its vicinity.

4.2.2 Cellular Inference Engine (CIE)

We consider a cellular automaton made of two finite arbitrary long layers of finite-state machines (cells), all identical. The operation of the system is synchronous, and the state of each cell at time $t+1$ depends only on the state of its vicinity cells, and on its own state at time t.

The behavior of a knowledge base can be represented by such a cellular automaton with two layers. A first layer, called CELFACT, represents the fact base, and a second layer, called CELRULE, represents the rule base. In each layer, the contents of a cell determines whether and how it participates in each inference step: at every step, a cell can be active or passive, can take part in the inference or not. We assume that there are l cells in the layer CELFACT, and r cells in the layer CELRULE.

Notations and definitions. The states of cells are composed of three parts: EF, IF and SF, and ER, IR and SR which are the input, internal state and output parts of the CELFACT cells, and of the CELRULE cells, respectively. The internal state of a CELFACT cell indicates the fact role: in our example $IF = 0$ corresponds to a fact of the form s_i, $IF = 1$ corresponds to a fact of the form $X_i = k$. In a CELRULE cell, IR can be used as a probability coefficient. We will not use it in our example.

Any cell i in the CELFACT layer with input $EF(i) = 1$ is regarded as representing an established fact. If $EF(i) = 0$, the represented fact has to be established. Any cell j of the CELRULE layer with input $ER(j) = 0$ is regarded as a candidate rule. When $ER(j) = 1$, the rule should not take part in the inference.

In order to illustrate the cellular automaton architecture and operation, let us consider the part of the graph (see Figure 4) obtained using the partitions $S_0 = (s_0)$, $S_1 = (s_1, s_2, s_3)$ and $S_2 = (s_4, s_5)$ of Figure 3. Figure 5 shows how the knowledge base extracted from this graph is represented by the automaton layers CELFACT and CELRULE.

Initially, all the cell inputs in the CELFACT layer are passive ($EF = 0$), except those which represent the initial fact base ($EF(1) = 1$).

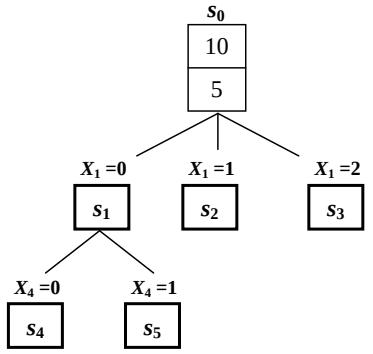


Fig. 4. The partitions S_0, S_1 and S_2

Rule j :		*Premise :*		*Conclusion :*
Rule 1	*if*	s_0	*then*	**(X_1= 0) and s_1**
Rule 2	*if*	s_0	*then*	**(X_1= 1) and s_2**
Rule 3	*if*	s_0	*then*	**(X_1= 2) and s_3**
Rule 4	*if*	s_1	*then*	**(X_4= 0) and s_4**
Rule 5	*if*	s_1	*then*	**(X_4= 1) and s_5**

a)

Fact i	
Fact 1	s_0
Fact 2	X_1= 0
Fact 3	s_1
Fact 4	X_1= 1
Fact 5	s_2
Fact 6	X_1= 2
Fact 7	s_3
Fact 8	X_4= 0
Fact 9	s_4
Fact 10	X_4= 1
Fact 11	s_5

EF	*IF*	*SF*
1	*0*	*0*
0	*1*	*0*
0	*0*	*0*
0	*1*	*0*
0	*0*	*0*
0	*1*	*0*
0	*0*	*0*
0	*1*	*0*
0	*0*	*0*
0	*1*	*0*
0	*0*	*0*
CELFACT		

Rule j	
Rule 1	R_1
Rule 2	R_2
Rule 3	R_3
Rule 4	R_4
Rule 5	R_5

ER	*IR*	*SR*
0	*1*	*1*
0	*1*	*1*
0	*1*	*1*
0	*1*	*1*
0	*1*	*1*
CELRULE		

b)

Fig. 5. a) Knowledge base; b) Initial cellular automaton configuration

Let R_E and R_S be the input and the output incidence matrices, respectively:

- the input relation, noted iR_Ej, is formulated as follows: $\forall i \in [1, l]$, $\forall j \in [1, r]$, if (fact $i \in$ Premise of rule j) then $R_E(i, j) \leftarrow 1$.
- the output relation, noted iR_Sj, is formulated as follows: $\forall i \in [1, l]$, $\forall j \in [1, r]$, if (fact $i \in$ Conclusion of rule j) then $R_S(i, j) \leftarrow 1$.

The incidence matrices R_E and R_S represent the input/output relation of the facts and are used in forward chaining. One can also use R_E as output relation and R_S as input relation for backward chaining. Notice that no cell in the vicini-

ty of a CELFACT (or CELRULE) cell belong to the CELFACT (or CELRULE) layer.

R_E	R_1	R_2	R_3	R_4	R_5
s_0	1	1	1		
X_1= 0					
s_1				1	1
X_1= 1					
s_2					
X_1= 2					
s_3					
X_4= 0					
s_4					
X_4= 1					
s_5					

R_S	R_1	R_2	R_3	R_4	R_5
s_0					
X_1= 0	1				
s_1	1				
X_1= 1		1			
s_2		1			
X_1= 2			1		
s_3			1		
X_4= 0				1	
s_4				1	
X_4= 1					1
s_5					1

Fig. 6. Input/output incidences matrices

Finally, since there are l cells in the layer CELFACT, EF, IF and SF will be considered as l-dimensional vectors (EF, IF, $SF \in \{0,1\}^l$). Similarly, since there are r cells in the layer CELRULE, ER, IR and SR will be considered as r-dimensional vectors (ER, IR, $SR \in \{0,1\}^r$). Figure 7 shows the general outline of our cellular automaton.

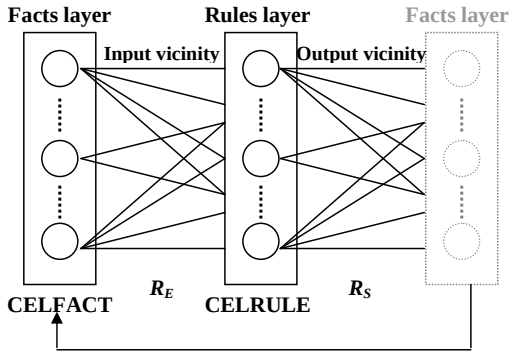


Fig. 7. Cellular automaton for systems of inference

4.2.3 CIE Transitions Functions

Given a goal fact, the basic cycle of an inference engine, in forward chaining, traditionally operates as follows:

1. Search for applicable rules (evaluation and selection);
2. Choose one of these rules, for example R (filtering);
3. Apply and add the conclusion part of R to the fact base (execution).

The cycle is repeated until the goal fact is added to the fact base, or stops when no rule is applicable.

The cellular automaton dynamics implements the CIE module (see Figure 1) as a cycle of an inference engine made up of two local transitions functions δ_{fact} and δ_{rule}, where δ_{fact} corresponds to the evaluation, selection and filtering phases, and δ_{rule} corresponds to the execution phase.

- δ_{fact} transition function:

$$(EF, IF, SF, ER, IR, SR) \rightarrow \delta_{fact}(EF, IF, EF, ER + (R_E^T \cdot EF), IR, SR)$$

- δ_{rule} transition function:

$$(EF, IF, SF, ER, IR, SR) \rightarrow \delta_{rule}(EF + (R_S \cdot ER), IF, SF, ER, IR, \overline{ER})$$

where the matrix R_E^T is the transpose of R_E.

We consider G_0 as the initial cellular automaton configuration (see Figure 5) and the $\Delta = \delta_{rule} \circ \delta_{fact}$, as a global transition function: $\Delta(G_0) = G_1$ if $G_0 \rightarrow \delta_{fact} G'_0$ and $G'_0 \rightarrow \delta_{rule} G_1$. Let $G = \{G_0, G_1, \ldots, G_q\}$ be the configuration set of our cellular automaton. The automaton evolution in discrete time steps from one generation to the next is defined by the configuration sequence G_0, G_1, ..., G_q, where $G_{i+1} = \Delta(G_i)$.

As an example, let us try first to establish the fact s_4 with the knowledge base of Figure 5 and without using the cellular principle. Figure 8 unrolls the forward chaining according to various modes:

- width synchronous mode, where all the candidate rules are triggered;
- width asynchronous mode;
- depth asynchronous mode.

Using our cellular automaton principle now, Figure 9 presents the two layers, CELFACT and CELRULE, after evaluation, selection and filtering in synchronous mode with the first transition law, δ_{fact}. After the application of the second transition law, δ_{rule}, we obtain the configuration G_1(see Figure 10).

Δ constitutes a forward total transition law which transforms iteratively our cellular automaton from an initial configuration into a final configuration (see Figure 11).

Similarly, and with the same transition functions, our cellular machine can carry out inferences in backward chaining. The only modification which should be made consists in permutating, in the transition laws, the incidences matrices R_E and R_S.

4.2.4 Induction Graph Initialization by Cellular Automaton (COG)

For the construction of the induction graph, with the SIPINA method, and using sample Ω_a, we start the symbolic treatment of the CELFACT and CELRULE layers.

Mode 1 : width synchronous

Cycle	Set fact base evolution	Rules
1	$\{s_0\} \cup \{(X_1{=}0), (X_1{=}1), (X_1{=}2), \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3\}$	1, 2, 3
2	$\{s_0, (X_1{=}0), (X_1{=}1), (X_1{=}2), \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3\} \cup \{(X_4{=}0), (X_4{=}1), \boldsymbol{s}_4, \boldsymbol{s}_5\}$	4, 5

Mode 2 : width asynchronous

Cycle	Set fact base evolution	Rules
1	$\{s_0\} \cup \{(X_1{=}0), \boldsymbol{s}_1\}$	1
2	$\{s_0, (X_1{=}0), s_1\} \cup \{(X_1{=}1), \boldsymbol{s}_2\}$	2
3	$\{s_0, (X_1{=}0), s_1, (X_1{=}1), s_2\} \cup \{(X_1{=}2), \boldsymbol{s}_3\}$	3
4	$\{s_0, (X_1{=}0), s_1, (X_1{=}1), s_2, (X_1{=}2), \boldsymbol{s}_3\} \cup \{(X_4{=}0), \boldsymbol{s}_4\}$	4

Mode 3 : depth asynchronous

Cycle	Set fact base evolution	Rules
1	$\{s_0\} \cup \{(X_1{=}0), \boldsymbol{s}_1\}$	1
2	$\{s_0, (X_1{=}0), \boldsymbol{s}_1\} \cup \{(X_4{=}0), \boldsymbol{s}_4\}$	4

Fig. 8. Forward chaining with various modes

Fact i		*EF*	*IF*	*SF*
Fact 1	$\boldsymbol{s_0}$	**1**	0	**1**
Fact 2	$\boldsymbol{X_1}$**= 0**	0	1	0
Fact 3	$\boldsymbol{s_1}$	0	0	0
Fact 4	$\boldsymbol{X_1}$**= 1**	0	1	0
Fact 5	$\boldsymbol{s_2}$	0	0	0
Fact 6	$\boldsymbol{X_1}$**= 2**	0	1	0
Fact 7	$\boldsymbol{s_3}$	0	0	0
Fact 8	$\boldsymbol{X_4}$**= 0**	0	1	0
Fact 9	$\boldsymbol{s_4}$	0	0	0
Fact 10	$\boldsymbol{X_4}$**= 1**	0	1	0
Fact 11	$\boldsymbol{s_5}$	0	0	0
		CELFACT		

Rule j		*ER*	*IR*	*SR*
Rule 1	$\boldsymbol{R_1}$	**1**	1	1
Rule 2	$\boldsymbol{R_2}$	**1**	1	1
Rule 2	$\boldsymbol{R_3}$	**1**	1	1
Rule 4	$\boldsymbol{R_4}$	0	1	1
Rule 5	$\boldsymbol{R_5}$	0	1	1
		CELRULE		

Fig. 9. Configuration obtained with δ_{fact}

Fact i		*EF*	*IF*	*SF*
Fact 1	$\boldsymbol{s_0}$	**1**	0	**1**
Fact 2	$\boldsymbol{X_1}$**= 0**	**1**	1	0
Fact 3	$\boldsymbol{s_1}$	**1**	0	0
Fact 4	$\boldsymbol{X_1}$**= 1**	**1**	1	0
Fact 5	$\boldsymbol{s_2}$	**1**	0	0
Fact 6	$\boldsymbol{X_1}$**= 2**	**1**	1	0
Fact 7	$\boldsymbol{s_3}$	**1**	0	0
Fact 8	$\boldsymbol{X_4}$**= 0**	0	1	0
Fact 9	$\boldsymbol{s_4}$	0	0	0
Fact 10	$\boldsymbol{X_4}$**= 1**	0	1	0
Fact 11	$\boldsymbol{s_5}$	0	0	0
		CELFACT		

Rule j		*ER*	*IR*	*SR*
Rule 1	$\boldsymbol{R_1}$	**1**	1	**0**
Rule 2	$\boldsymbol{R_2}$	**1**	1	**0**
Rule 2	$\boldsymbol{R_3}$	**1**	1	**0**
Rule 4	$\boldsymbol{R_4}$	0	1	1
Rule 5	$\boldsymbol{R_5}$	0	1	1
		CELRULE		

Fig. 10. Configuration $G_1 = \Delta(G_0)$ obtained with $\delta_{fact}(G_0)$ and $\delta_{rule}(G_0)$

Fact i	
Fact 1	s_0
Fact 2	$X_1 = 0$
Fact 3	s_1
Fact 4	$X_1 = 1$
Fact 5	s_2
Fact 6	$X_1 = 2$
Fact 7	s_3
Fact 8	$X_4 = 0$
Fact 9	s_4
Fact 10	$X_4 = 1$
Fact 11	s_5

EF	IF	SF
1	0	1
1	1	1
1	0	1
1	1	1
1	0	1
1	1	1
1	0	1
1	1	0
1	0	0
1	1	0
1	0	0
CELFACT		

Rule j	
Rule 1	R_1
Rule 2	R_2
Rule 2	R_3
Rule 4	R_4
Rule 5	R_5

ER	IR	SR
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
CELRULE		

Fig. 11. Final configuration G_2 after two synchronous iterations

To initialize the cellular automaton, the COG module (see Figure 1) uses three procedures and proceeds as follows:

- set the measure of uncertainty (Quadratic or Shannon);
- set the parameters: λ, μ and the initial partition S_0;
- $i \leftarrow 2$; (row index)
- $j \leftarrow 1$; (column index)
- $(EF, IF, SF)[1] \leftarrow (0, 0, 0)$; ($s_0$ initialization, see Figure 13)
- use the SIPINA method to go from partition S_i to S_{i+1}.

For going from partition S_i to S_{i+1} the algorithm proceeds as follows:

```
Repeat
If Fusion possible then Fcell
           else
           if Fusion-splitting possible then FScell
                     else
                     if Splitting then Scell
                     else stop
```

In order to define the Fcell, FScell and Scell procedures, let us consider the part of the induction graph (Figure 12) obtained using the partitions $S_0 = (s_0)$, $S_1 = (s_1, s_2, s_3)$, $S_2 = (s_4)$ and $S_3 = (s_5, s_6, s_7)$.

Scell: The partition S_0 includes one element s_0. Assuming that splitting S_0 is possible, the Scell procedure finds in CELFACT the index of root s_0, noted i_{root} ($i_{root} = 1$ in this case) for initializing the input incidence matrix ($R_E[i_{root}, j] \leftarrow 1$), and repeats, for each modality of the attribute X_1, the following steps: create two cells in the CELFACT layer, create one cell in the CELRULE layer, and initializes the output incidence matrix R_S; for example (see Figure 13), $(EF, IF, SF)[2] \leftarrow (0, 1, 0)$ for a fact of the form $X_1 = 0$, $(EF, IF, SF)[3] \leftarrow$

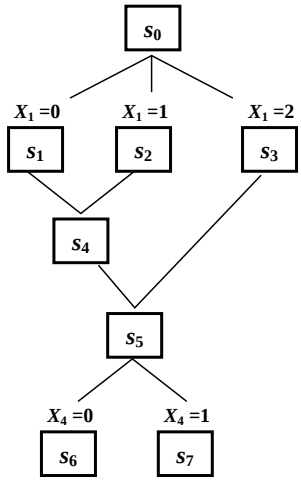
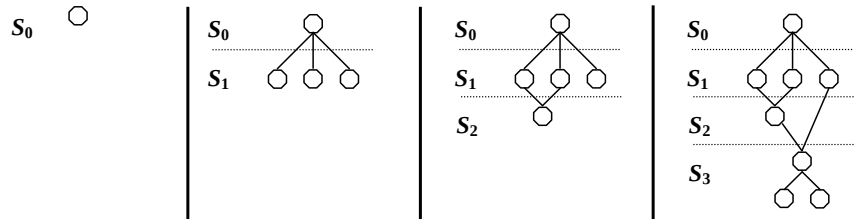


Fig. 12. The partitions S_0, S_1, S_2 and S_3

$(0,0,0)$ for a fact of the form s_1 and $(ER, IR, SR)[1] \leftarrow (0,1,1)$ for a rule of the form `if` s_0 `then` $(X_1 = 0)$ `and` s_1. For each new cell creation in the CELRULE layer, Scell initializes the output incidence matrix ($R_S[2,1] \leftarrow 1$ and $R_S[3,1] \leftarrow 1$).

Fcell: The partition S_1 includes three elements s_1, s_2 and s_3. Assuming that the fusion S_1 into S_2 is possible, the Fcell procedure, first searches the indices of s_1 and s_2 in CELFACT (or i_1 and i_2, $i_1 = 3$ and $i_2 = 5$ in Figure 13) and initializes the input incidence matrix (for $j = 4$, $R_E[3,j] \leftarrow 1$ and $R_E[5,j] \leftarrow 1$). Then, Fcell creates the s_4 node in CELFACT with a new row index ($(EF, IF, SF)[8] \leftarrow (0,0,0)$), initializes the output incidence matrix ($R_S[8,j] \leftarrow 1$) and creates a new cell in CELRULE ($(ER, IR, SR)[4] \leftarrow (0,1,1)$) for the rule `if` s_1 `and` s_2 `then` s_4.

FScell: Using the Fcell and Scell procedures, FScell completes the CELFACT and CELRULE layers and creates the S_3 partition with nodes s_5, s_6 and s_7. The FScell procedure first uses Fcell for fusion and, then if splitting is possible, uses Scell starting from the node obtained by fusion. In our case (see Figure 13), Fcell first creates the s_5 node in CELFACT with a new row index equal to 9 and the rule $(ER, IR, SR)[5] \leftarrow (0,1,1)$ in CELRULE layer (for the rule *if* s_3 and s_4 *then* s_5). Then, Scell creates, stating from s_5, four new cells in the CELFACT layer ($X_4 = 0$, s_6, $X_4 = 1$ and s_7) and two cells in the CELRULE layer ($(ER, IR, SR)[6]$ for `if` s_5 `then` $(X_4 = 0)$ `and` s_6 and $(ER, IR, SR)[7]$ for `if` s_5 `then` $(X_4 = 1)$ `and` s_7.



CELFACT	CELRULE
(0,0,0) [1]	

Initialization

CELFACT	CELRULE
(0,0,0) [1]	**(0,1,1) [1]**
(0,1,0) [2]	**(0,1,1) [2]**
(0,0,0) [3]	**(0,1,1) [3]**
(0,1,0) [4]	
(0,0,0) [5]	
(0,1,0) [6]	
(0,0,0) [7]	

S c e l l

CELFACT	CELRULE
(0,0,0) [1]	(0,1,1) [1]
(0,1,0) [2]	(0,1,1) [2]
(0,0,0) [3]	(0,1,1) [3]
(0,1,0) [4]	**(0,1,1) [4]**
(0,0,0) [5]	
(0,1,0) [6]	
(0,0,0) [7]	
(0,0,0) [8]	

F c e l l

CELFACT	CELRULE
(0,0,0) [1]	(0,1,1) [1]
(0,1,0) [2]	(0,1,1) [2]
(0,0,0) [3]	(0,1,1) [3]
(0,1,0) [4]	(0,1,1) [4]
(0,0,0) [5]	**(0,1,1) [5]**
(0,1,0) [6]	**(0,1,1) [6]**
(0,0,0) [7]	**(0,1,1) [7]**
(0,0,0) [8]	
(0,0,0) [9]	
(0,1,0) [10]	
(0,0,0) [11]	
(0,1,0) [12]	
(0,0,0) [13]	

F S c e l l

Fig. 13. Cellular automaton initialization

4.3 Induction Graph Optimization

In order to illustrate graph optimization and rules generation by the cellular method using induction graphs, Figure 14 shows some possible useless splitting fusions cases. The majority class is associated with each terminal node in the graphs of Figure 14 a). We obtain as many rules as there are terminal nodes and, in each rule, as many conjunctions as there are branches back to the root.

For example, the induction graph G_3 of Figure 14 a) can be expressed by the following set of rules:

R_1: `if` $((X_3 = 0)$ `and` $(X_2 = 0)$ `and` $(X_1 = 2))$ `or` $((X_3 = 1)$ `and` $(X_2 = 0)$ `and` $(X_1 = 2))$ `then majority class of` s_8.

R_2: `if` $((X_2 = 1)$ `and` $(X_1 = 2))$ `then majority class of` s_5.

R_3: `if` $(X_1 = 1)$ `then majority class of` s_2.

R_4: `if` $(X_1 = 0)$ `then majority class of` s_1.

In knowledge discovery from database, the rules are generated from a training sample and have a double objective of characterizing the classes of concepts, and assigning a class to an example whose class is unknown. In the production rules which we wish to generate, the condition is a conjunction of elementary propositions made of an attribute, an operator ($=$, $\neq$, $<$, $>$, ...) and an attribute value. The conclusion consists of a particular proposition where the attribute relates to the class

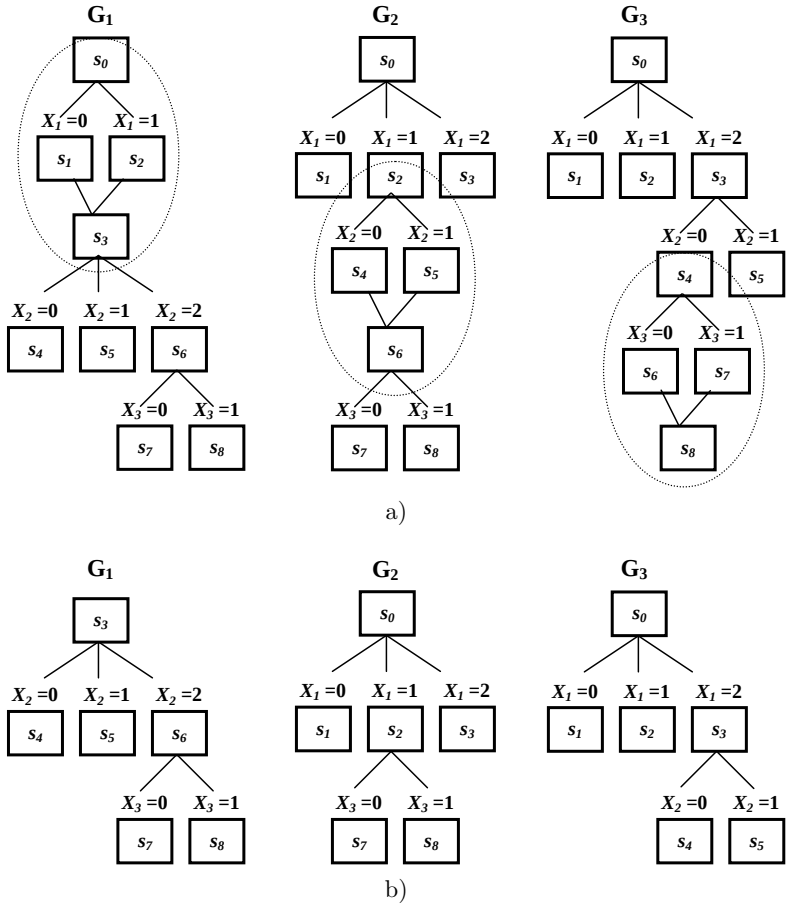
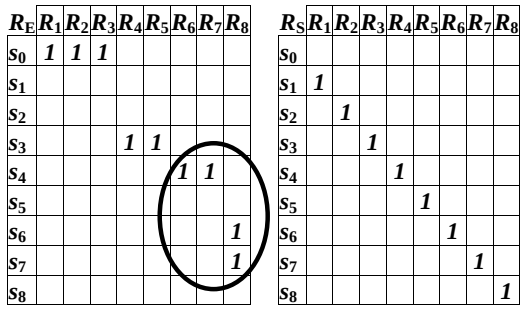


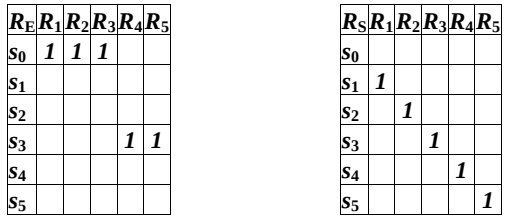
Fig. 14. Some possible useless splitting-fusion cases

(for example diabetes I or II). It is possible to associate with each rule a coefficient which defines the certainty, or probability, with which a class is predicted. After data exploration, the cellular automaton assists the SIPINA method to generate a decision graph. This graph is represented using only R_E because, for such a type of graph, the output matrix R_S is elementary and does not even require an internal representation.

Now, our simplification process is based on elementary propositional calculus and uses the matrix R_E. We show here the rules generation with detection of useless splitting-fusion operations. This corresponds to seeking and eliminating propositions of the from $(A + \overline{A})$. By exploring R_E our algorithm seeks the columns with a sum greater than 1 (fusion rules). For each fusion rule, the algorithm checks if the nodes taking part in this fusion were produced from the same node. Figure 15 shows the G_3



a)



b)

Fig. 15. R_E and R_S of G_3 graph before and after optimization

graph matrices R_E and R_S before and after the optimization procedure. We notice that the algorithm has detected and eliminated the proposition $(X_3 = 0) + \overline{(X_3 = 0)}$ which is equivalent, according to the R_E matrix, to $s_4 = s_6 + s_7$, $s_6 = s_8$ and $s_7 = s_8$, that is $s_4 = s_8$.

4.4 Generation of Conjunctive Rules

To automatically generate conjunctive rules we use same δ_{fact} and δ_{rule} transition functions with the permutation of R_E and R_S. We suppose that all the facts of the form $X_i = k$ are established (EF=1). Going from the terminal nodes back to the root, s_0, and launching the cellular inference engine (CIE) in back chaining with a depth asynchronous mode imposed by the form of R_S, we follow the steps shown in Figure 16.

We proceed in the same way with the graph of Figure 3 and after the elemination of the X_4 variable, we obtain the following conjunctive rules:

R_1: if $(X_1 = 1)$ and $(X_5 = 1)$ then majority class of s_9

R_2: if $(X_1 = 2)$ and $(X_5 = 1)$ then majority class of s_9

R_3: if $(X_1 = 1)$ and $(X_5 = 0)$ and $(X_{10} = 0)$ then majority class of s_{11}

R_4: if $(X_1 = 2)$ and $(X_5 = 0)$ and $(X_{10} = 0)$ then majority class of s_{11}

R_5: if $(X_1 = 0)$ and $(X_{10} = 0)$ then majority class of s_{11}

Cycle number	Fact base evolution	Rule number
1	$\{s_5, (X_2{=}1)\} \cup \{s_3\}$	5
2	$\{s_5, (X_2{=}1), s_3, (X_1{=}2)\} \cup \{s_0\}$	3
3	$\{s_4, (X_2{=}0)\} \cup \{s_3\}$	4
4	$\{s_4, (X_2{=}0), s_3, (X_1{=}2)\} \cup \{s_0\}$	3
5	$\{s_2, (X_1{=}1)\} \cup \{s_0\}$	2
6	$\{s_1, (X_1{=}0)\} \cup \{s_0\}$	1

Rule base :
if $(X_2{=}1)$ ***and*** $(X_1{=}2)$ ***then majority class of*** s_5
if $(X_2{=}0)$ ***and*** $(X_1{=}2)$ ***then majority class of*** s_4
if $(X_1{=}1)$ ***then majority class of*** s_2
if $(X_1{=}0)$ ***then majority class of*** s_1

Fig. 16. Generation of conjunctive rules

R_6: `if` $(X_1 = 1)$ `and` $(X_5 = 0)$ `and` $(X_{10} = 1)$ `then majority class of` s_{12}
R_7: `if` $(X_1 = 2)$ `and` $(X_5 = 0)$ `and` $(X_{10} = 1)$ `then majority class of` s_{12}
R_8: `if` $(X_1 = 0)$ `and` $(X_{10} = 1)$ `then majority class of` s_{12}

The representation of this knowledge base by the cellular machine is illustrated in Figure 17. Upon completion of this process, the cellular machine is ready to launch the validation phase. By using the same guiding principle of an inference engine and the same δ_{fact} and δ_{rule} transition functions, the cellular automaton advances from a configuration to the next, for finally generating the set Ω_e.

4.5 Validation Using the Cellular Automaton

Let us suppose that our test sample Ω_t (Table 4) is composed of 10 diabetic patients belonging to two classes 1 and 2, where class 1, diabetes of type I, is the majority class of s_9 and s_{11}, and class 2, diabetes of type II, is the majority class of s_{12}. Figure 18 summarizes the validation of individual ω_1.

After the rules optimization and generation by the cellular machine (SIPINA coupled with CASI) and using the same δ_{fact} and δ_{rule} transition functions, the validation has been tested on several data bases of larger sizes: one with 150 individuals containing only continuous variables, a second one of 2201 individuals containing only discrete variables, and a third one of 2000 individuals containing the two types of variables.

These experimentations have shown that the new graph representation and optimization leave the classification success unchanged, with a success rate of approximately 85 %, whereas this reduces the graph size by more than 50 %, as well as the validation time and the number of descriptive variables. These results show not only that one can generate in an optimal way the simple conjunctive production rules starting from an induction graph, but also encourage the exploitation of this

Fact i	
Fact 1	X_1=0
Fact 2	X_1=1
Fact 3	X_1=2
Fact 4	X_5=0
Fact 5	X_5=1
Fact 6	X_{10}=0
Fact 7	X_{10}=1
Fact 8	Class s_9
Fact 9	Class s_{11}
Fact 10	Class s_{12}

EF	*IF*	*SF*
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
CELFACT		

Rule j	
Rule 1	R_1
Rule 2	R_2
Rule 3	R_3
Rule 4	R_4
Rule 5	R_5
Rule 6	R_6
Rule 7	R_7
Rule 8	R_8

ER	*IR*	*SR*
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
CELRULE		

R_E	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
X_1=0					1			1
X_1=1	1		1			1		
X_1=2		1		1			1	
X_5=0			1	1		1	1	
X_5=1	1	1						
X_{10}=0			1	1	1			
X_{10}=1						1	1	1
Class s_9								
Class s_{11}								
Class s_{12}								

R_S	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
X_1=0								
X_1=1								
X_1=2								
X_5=0								
X_5=1								
X_{10}=0								
X_{10}=1								
Class s_9	1	1						
Class s_{11}			1	1	1			
Class s_{12}						1	1	1

Fig. 17. Knowledges base of the figure 3 induction graph

new technique to optimize the size and time. This cellular automaton is now integrated with the SIPINA method in order to have a complete autonomous device for automated retrieval and validation of knowledge starting from data.

Ω_t	**CLASS**	X_1	X_5	X_{10}
ω_1	1	1	1	0
ω_2	1	2	1	1
ω_3	1	1	0	0
ω_4	1	2	0	0
ω_5	1	0	1	0
ω_6	1	1	1	1
ω_7	2	1	0	1
ω_8	2	2	0	1
ω_9	2	0	0	1
ω_{10}	2	0	1	1

Table 4. Example of test sample

ω_{10}	ω_9	ω_8	ω_7	ω_6	ω_5	ω_4	ω_3	ω_2	ω_1	➔ ➔
1	1	0	0	0	1	0	0	0	0	X_1=0
0	0	0	1	1	0	0	1	0	1	X_1=1
0	0	1	0	0	0	1	0	1	0	X_1=2
0	1	1	1	0	0	1	1	0	0	X_5=0
1	0	0	0	1	1	0	0	1	1	X_5=1
0	0	0	0	0	1	1	1	0	1	X_{10}=0
1	1	1	1	1	0	0	0	1	0	X_{10}=1
0	0	0	0	1	1	1	1	1	1	*Class 1*
1	1	1	1	0	0	0	0	0	0	*Class 2*

EF	*IF*	*SF*
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
CELFACT		

ER	*IR*	*SR*
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
CELRULE		

ω_{10}	ω_9	ω_8	ω_7	ω_6	ω_5	ω_4	ω_3	ω_2	ω_1	➔ ➔
1	1	0	0	0	1	0	0	0	0	X_1=0
0	0	0	1	1	0	0	1	0	1	X_1=1
0	0	1	0	0	0	1	0	1	0	X_1=2
0	1	1	1	0	0	1	1	0	0	X_5=0
1	0	0	0	1	1	0	0	1	1	X_5=1
0	0	0	0	0	1	1	1	0	1	X_{10}=0
1	1	1	1	1	0	0	0	1	0	X_{10}=1
0	0	0	0	1	1	1	1	1	1	*Class 1*
1	1	1	1	0	0	0	0	0	0	*Class 2*

EF	*IF*	*SF*
0	1	0
1	1	**1**
0	1	0
0	1	0
1	1	**1**
0	1	0
0	1	0
1	1	0
0	1	0
CELFACT		

ER	*IR*	*SR*
1	1	**0**
0	1	1
0	1	1
0	1	1
0	1	1
0	1	1
CELRULE		

Fig. 18. Validation of ω_1 by the cellular automaton

5 CONCLUSION

In the context of our guiding example, the diabetes data base, our results confirm our assumption that it is possible to determine the relevant exogenous variables which characterize the various types of diabetes. In this context, with a confirmation of these results on larger examples, we could contribute to the construction of a cellular expert system for the monitoring of diabetics patients and the prediction of diabetes types.

Our new principle of representation also enabled us to test the performance of our cellular automaton within the framework of real applications and not only on artificial problems. We have shown that the induction graphs, illustrated by the SIPINA method, constitute a very satisfactory tool for the extraction of knowledge starting from data, by proposing a function of effective and explanatory classification. The graph optimization and the automatic rules generation, by our automaton, highlighted the possibility of feeding the knowledge base of a cellular expert system and raised the problem of rules formalization and simplification.

Two competing objectives led us to propose a cellular automaton for the optimization, generation, representation and use of a knowledge base. Indeed, we not only wished to have an optimal knowledge base, but we also wished to improve the construction of an expert system by proposing a new cellular technique. The

advantages of our method based on cellular automaton can be summarized as follows:

- Information-gathering and ordering is simple, in the form of binary matrices requiring minimal pre-processing.
- The transition functions are easy to use, of low complexity, effective and robust regarding extreme values. Moreover, they are well adapted to situations with many attributes.
- The results are simple to include in and use by an expert system.
- The cellular model amounts to a simple set of transition functions and production rules, which not only make it possible to describe the problem at hand but also to build a classification function for class prediction.
- The incidence matrix, R_E, facilitates the rules transformation into Boolean equivalent expressions and makes it possible, thereafter, to rely on elementary boolean algebra to test other simplifications.

In order to make the SIPINA method more general, by taking into account all the data types, our cellular machine makes it possible to extend it and to take into account the degree of certainty of a rule. This could be given either in a possibilist way by experts of the field, or by an automatic calculation, and would be represented in our machine by the rule internal state IR. Moreover, in order to use the SIPINA method as a KDD method on gigantic size data bases, it appears interesting to use the cellular automaton principle in order to take advantage of the possibilities of parallelism within the SIPINA algorithms.

Taking into account the incremental and evolutionary character of the knowledge extraction, in many fields, in particular in that of the diabetes characterization, it appears necessary to equip the same device with rules inference, with capacities to build its own expertise using various types of numerical tools and symbolic systems. We think that a prospect for the current and future developments is the construction of cellular expert systems, allowing the direct introduction of rules given by the expert as well as their production starting from a data base, by symbolic training systems containing induction graphs. Such systems would make it possible to benefit from all the knowledge sources and forms available, by handling various knowledge representations through several reasoning modes (front, back or mixed chaining).

Acknowledgements

I would like to express my gratitude to Philippe Jorrand who has significantly and decisively contributed to improving the written presentation of this work.

REFERENCES

[1] Atmani, B.—Beldjilali, B.: The Invocation of Front and Back Chaining Rules on Cellular Automaton for Inference Systems Containing Rules. International conference in data processing, University Of Amman, Jordani, July 8–9, 1997.

[2] Breiman, L.—Friedman, J. H.—Olshen, R. A.—Stone, C. J.: Classification and Regression Trees. Technical report. Wadsworth International, Monterey, CA, 1984.

[3] Buen-Rodriguez, P. R.—Morales, E. F.—Vadera, S.: RuLess: A Method for the Acquisition and Simplification of Rules. Proceedings of the Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence, Springer-Verlag, London, UK, 2000, pp. 272–283.

[4] Cantu-Paz, E.—Kamath, C.: Induction Oblique Decision Tree with Evolutionary Algorithms. IEEE Transaction on Evolutionary Computation, Vol. 7, 2003, No. 1, pp. 54–69.

[5] Chopard, B.—Luthi, P. O.—Queloz, P. A.: Cellular Automata Model of Car Traffic in a Two-Dimensional Street Network. Physica A. May 1996.

[6] Denis, F.—Gilleron, R.: Apprentissage Partir d'Exemples. Technical report. Grappa – University of Lille 3, 1999.

[7] Duch, W.—Adamczak, R.—Grabczewski, K.: Methodology of Extraction, Optimization and Application of Logical Rules. Intelligent Information Systems VIII Proceedings of the Workshop held in Ustroń, Poland, June 14–18 , 1999.

[8] Duhamel, A.—Picavet, M.—Devos, P.—Beuscart, R.: From Data Collection to Knowledge Data Discovery – A Medical Application of Data Mining. Studies in Health Technology and informatics, Vol. 84, 2001, pp. 1329–1333.

[9] Fayyad, U.—Shapiro, G. P.—Smyth, P.: The KDD Process for Extraction Useful Knowledge from Volumes Data. Communication of the ACM, 1996.

[10] Ganascia, J. G.: Charade - A Study of the Learning Bias Semantics. European Conference on Artificial Intelligence, ECAI, Munich, 1988.

[11] Herr, A.—Klomp, N. I.—Atkinson, J. S.: Identification of Bat Echolocation Calls Using a Decision Tree Classification System. Complexity international, Vol. 4, 1997.

[12] Glymour, C.—Madigan, D.—Pregibon, D.—Smyth, P.: Statistical Inference and Data Mining. Communication of the ACM, Vol. 39, 1996, No. 11, pp. 35–41.

[13] Kernerl, B. S.—Klenov, S. L.—Wolf, D. E.: Cellular Automata Approach to Three-Phase Traffic Theory. Physica A. Nov. 2002.

[14] Kodratoff, Y.: The Extraction of Knowledge from Data, A New Topic for the Scientific Research. Magazine electronic READ, 1997.

[15] Kohavi, R.—Quinlan, J.: Decision Tree Discovery. In Handbook of Data Mining and Knowledge Discovery, Klosgen and Zytkow, Editors, 2002, pp. 267–276.

[16] Lebart, L.—Morineau, A.—Piron, M.: Statistique Exploratoire Multidimensionnelle. Dunod, 2000.

[17] Lee, H.—Ong, H.: Visualization Support for Data Mining. IEEE Expert, Vol. 11, 1996, No. 5, pp. 69–75.

[18] Lee, I. N.—Lee, S. C.—Embrechts, M. J.: Important Variable Selection Techniques with Multiple Solutions for Medical Information Applications. Medical Informatics and the Internet in Medicine, Vol. 27, Dec. 2002, No. 4, pp. 253–266.

[19] Lee, S. J.—Sian, K.: A Review of Data Mining Techniques. MCB UP Ltd, Industrial Management and Data Systems, Vol. 11, Feb. 2001, pp. 41–46.

[20] Lefebure, R.—Venturi, G.: Data Mining. Paris, EYROLLES, 2001.

[21] Liao, S. C.—Lee, I. N.: Appropriate Medical Data Categorization for Data Mining Classification Techniques. Medical Informatics and the Internet in Medicine, Vol. 27, Jan. 2002, No. 1, pp. 59–67.

[22] Lu, H.—Setiono, R.—Liu, H.: Effective Data Mining Using Neural Networks. IEEE Transaction on Knowledge and Data Engineering, Vol. 8, 1996, No. 6, pp. 957–961.

[23] Nandi, S.—Kar, B. K.—Chaudhuri, P. P.: Theory and Applications of Cellular Automata in Cryptography. IEEE Transaction on Computers, Vol. 43, Dec. 1994, No. 12, pp. 1346–1357.

[24] Olaru, C.—Wehenkel, L.: A Complete Fuzzy Decision Tree Technique. Fuzzy Set and Systems, Vol. 138, 2003, No. 2.

[25] Quinlan, J. R.: Induction of Decision Trees. Machine Learning, 1986.

[26] Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[27] Rabaseda, S.—Zighed, D. A.: Generation and Simplification of Rules in Graphs of Induction. Acts of the 25th Symposium of the Economic Structures, Econometrics and Data Processing, 1996, p. 7.

[28] Rakotomalala, R.—Zighed, D. A.: Feschet, Characterization of Production Rules in a Process of Induction. Hermes Science Publication, Paris 1999.

[29] Schonfisch, B.—Roos, A.: Synchronous and Asynchronous Updating in Cellular Automata. Biosystems, Vol. 51, 1999, No. 3, pp. 123–143.

[30] Sebban, M.—Mokrousov, I.—Rastogi, N.—Sola, C.: A Data Mining Approach to Spacer Oligonucleotide Typing of Mycrobacterium Tuberculosis, Bioinformatics, Electronic Edition. Vol. 18, Feb. 2002, No. 2, pp. 235–243.

[31] Sirakoulis, G. C.—Karafyllidis, I.—Thanailakis, A.: A Cellular Automaton Methodology for the Simulation of Integrated Circuit Fabrication Processes. Future Generation Computer Systems, Vol. 18, Apr. 2002, No. 5, pp. 639–657.

[32] Wolf, D. E.: Cellular Automata for Traffic Simulations. Physica A, Vol. 263, Feb. 1999, No. 1, pp. 438–451.

[33] Wolfram, S.: Theory and Application of Cellular Automata. World Scientific 1986.

[34] Wolfram, S.: Cellular Automata and Complexity. Perseus Books Group, 2002.

[35] Zighed, D. A.: SIPINA for Windows, ver 2.5. Laboratory ERIC, University of Lyon, 1996.

[36] Zighed, D. A. —Rakotomalala, R.: Graphs of Induction, Training and Data Mining. Hermes Science Publication, 2000, pp. 21–23.

Baghdad Atmani graduated in 1991 from the Department of Computer Science in Oran (Algeria), and obtained his Master of Science Degree in the same department in 1996. He is currently a Ph. D. candidate in the Computer Science Department at the University of Oran. His research interests include knowledge discovery in databases, data mining, feature selection, neural networks, and cellular automata. A substantial part of his recent research was conducted in cooperation with the Laboratory of Informatics of Grenoble, France.

Bouziane Beldjilali received his Ph. D. degree in computer science from the University of Oran (Algeria), in 1996. He is a professor in the Computer Science Department at the University of Oran. His research interests include formal specifications, knowledge management, databases, artificial intelligence and automatic learning.